SliMOO: Interpretable Multi-Objective Evolutionary Search for LLM Depth Pruning

Guanchen Li, Yixing Xu, Xuanwu Yin, Dong Li, Emad Barsoum
Advanced Micro Devices, Inc. (AMD)
{guanchen.li,yixing.xu,xuanwu.yin,d.li,emad.barsoum}@amd.com

Abstract

As large language models (LLMs) grow more powerful, their scale makes them costly to serve. Depth pruning offers a practical route to efficiency by directly reducing memory and latency without hardware-specific support. However, we observe that transformer layers exhibit markedly different task-dependent layer contributions, whereas existing depth-pruning methods ignore this heterogeneity and thus tend to optimize for a narrow domain or benchmark rather than preserve broad capability. This suggests that LLM depth pruning should be formulated as a multi-objective optimization problem rather than a single-score ranking problem.

We propose **SliMOO**, an interpretable multi-objective evolutionary search framework for LLM depth pruning. SliMOO first uses single-layer removal probing to construct a task-layer importance map, which provides an interpretable view of task-shared and task-specific layers and serves as a search prior. It then performs a Pareto-aware evolutionary search for pruning masks, where candidates are generated under the guidance of the task-layer prior, scored by their deviation from the dense model on each task, and retained through NSGA-II environmental selection. Across the Llama-3.1 and Qwen3 model families, SliMOO consistently achieves stronger multi-task trade-offs than rule-based, greedy, and scalarized baselines. For example, on Llama-3.1-70B at 25% sparsity, it improves the math-domain average from 24.33% to 34.40% and the code-domain average from 35.03% to 39.14%. Code will be made available at `https://github.com/AMD-AGI/SliMOO`.

1 Introduction

Large language models (LLMs) have advanced rapidly in capability [1–3], but their growing scale makes their deployment increasingly costly. This tension has motivated extensive work on model compression, such as pruning, sparsity, and quantization [4–6]. Among these approaches, we focus on depth pruning, which removes entire transformer layers or sublayers from a pretrained model and therefore translates more directly into practical speedups on standard hardware, without requiring specialized sparse kernels or custom accelerators [7, 8].

Despite practical advantages, existing layer-pruning methods typically estimate layer importance with a single global score, derived from heuristics [9, 8], calibration loss [10], or scalarized search [11, 12]. This presumes a universal pruning order across capability domains, whereas we observe markedly different task-dependent contributions across transformer layers, suggesting that different capabilities rely on different depths. Consequently, a pruning strategy that appears successful by a headline metric may still fail to preserve broad capability: on Llama-3.1-70B at 25% sparsity, ShortGPT keeps MMLU, a knowledge reasoning benchmark, nearly unchanged (77.79 versus 78.51 for the dense model), yet sharply degrades GSM8K (math) from 80.82 to 12.74 and HumanEval (code) from 51.83 to 18.29. This cross-domain divergence indicates that depth pruning for LLMs is inherently a multi-objective optimization problem rather than a single-score ranking problem.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

To address this gap, we propose **SliMOO**[1], an interpretable multi-objective evolutionary search framework for LLM depth pruning. The key idea is to move from a single global pruning order to a task-aware multi-objective search over pruning masks. Concretely, SliMOO treats each task as a separate objective, defined by the KL divergence between the dense and pruned models' output distributions on the same task-specific inputs, and searches for masks that preserve strong multi-task trade-offs under a fixed sparsity budget. It also yields an interpretable view of which layers are shared across tasks and which are task-specific. SliMOO realizes this idea with two main contributions:

- We introduce **task-layer importance profiling (TLIP)**, which uses single-layer removal probing to quantify the task-wise contribution of individual layers. TLIP yields an interpretable task-layer importance map that identifies task-shared and task-specific layers, and provides the task-layer prior that guides the subsequent search.
- SliMOO performs a **Pareto-aware evolutionary search** over pruning masks. It generates candidate pruning masks through TLIP prior-guided perturbations and sparsity-preserving crossover, evaluates them on the task-wise objectives, and retains a diverse set of high-quality solutions through NSGA-II environmental selection, which combines non-dominated sorting with diversity preservation. This procedure enables SliMOO to identify pruning masks that achieve stronger multi-task trade-offs than single-score ranking methods.

Together, these components make SliMOO an effective and interpretable framework for LLM depth pruning. Across the Llama-3.1 and Qwen3 model families, SliMOO consistently delivers stronger multi-task trade-offs than rule-based, greedy, and scalarized baselines (*cf.* Figure 1). For example, on Llama-3.1-70B at 25% sparsity, SliMOO reduces the average language modeling perplexity by 1.46 relative to ShortGPT and is on par with EvoPress, while improving the math-, code-, and reasoning-domain averages by $21.55/10.07$, $23.69/4.11$, and $0.97/2.27$ relative to ShortGPT and EvoPress, respectively.

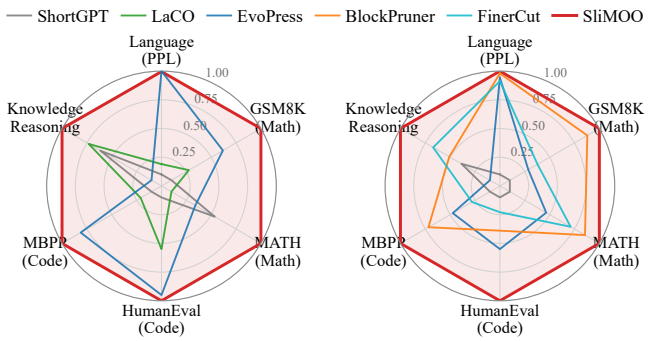


Figure 1: Normalized cross-domain trade-offs on Llama-3.1-70B. Scores are normalized within each setting; language modeling perplexity is inverted before normalization.

2 Related Work

Depth pruning for large language models. Recent work has explored LLM depth pruning to improve deployment efficiency. ShortGPT [8] ranks transformer layers using an activation cosine similarity score and removes globally redundant layers. LaCo [9] reduces depth through layer collapse (parameter merging) rather than direct deletion. EvoPress [11] applies evolutionary search to identify budget-aware compression configurations, while FinerCut [12] and BlockPruner [10] prune finer-grained attention/MLP sublayers using calibrated search or output-change signals. These methods demonstrate that substantial depth or structure can be removed, but they typically rely on a single global importance notion or a scalar objective. A related line of work skips computation dynamically at inference time through early exit or token-dependent routing, as in CALM [13], LayerSkip [14], and Mixture-of-Depths [15]. Such methods can reduce FLOPs or latency, but because they do not permanently remove layers, they do not reduce checkpoint size or memory footprint, and are often harder to integrate with static serving frameworks such as vLLM [16] and SGLang [17].

Multi-objective optimization. Multi-objective optimization (MOO) has a long history in optimization theory and algorithms, from early Pareto-based genetic algorithms [18, 19] to comprehensive treatments of multi-objective search [20]. NSGA-II [21] in particular established non-dominated sorting and diversity-preserving environmental selection as a standard template for approximating

[1] "Slim" refers to model slimming through depth pruning, and "MOO" stands for multi-objective optimization.

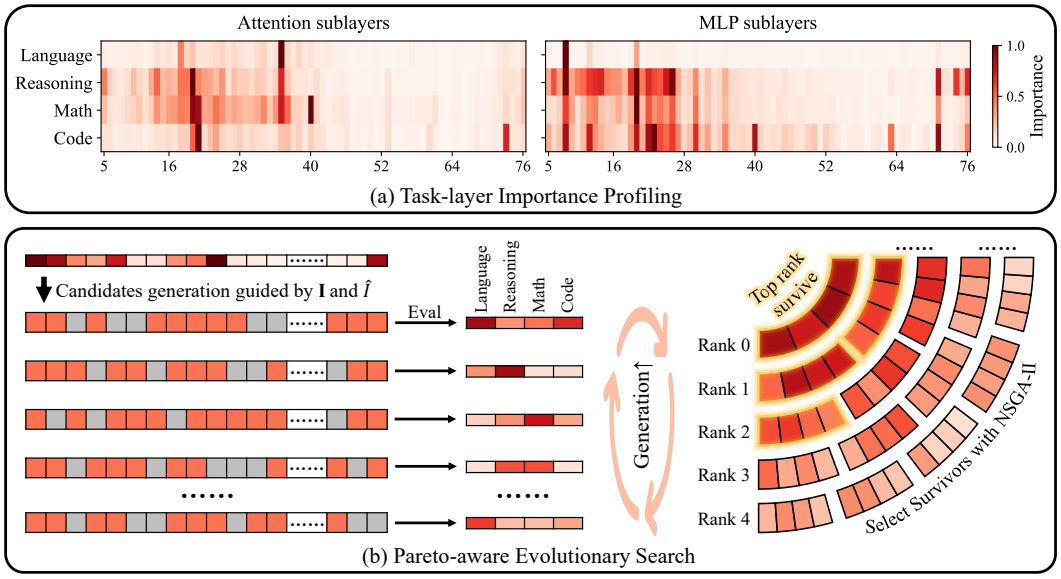


Figure 2: **Overview of SliMOO.** (a) TLIP profiles task-wise importance of layers/sublayers, yielding a task-layer importance map **I**. (b) Guided by the task-layer prior, SliMOO generates feasible pruning masks and uses NSGA-II to retain Pareto-competitive trade-offs under a target sparsity.

Pareto fronts. In deep learning, multi-objective search has also been applied to neural architecture search and deployment-aware model design to balance accuracy with resource constraints [22]. For model compression, MOSP [23] formulates unstructured LLM pruning as a multi-objective problem, using ADMM to identify shared and task-specific support that generate Pareto sparse models.

LLM interpretability and task sensitivity. A growing line of work investigates the functional roles of transformer components at multiple granularities, from layers and heads to neurons and parameters, using probing, ablation, and intervention-based analysis. Probing studies showed that different linguistic phenomena become recoverable at different depths [24]. Some recent work examines which layers are globally important [25] and which internal units become specialized for different domains in multi-task settings [26]. Taken together, these studies suggest that contributions to model behavior are unevenly distributed across different components.

Positioning of SliMOO. SliMOO is motivated by a key limitation of existing depth-pruning methods: they do not explicitly account for the task-specific functional roles of layers. Unlike MOSP-style intra-weight support optimization, SliMOO operates on layer/sublayer supports in a discrete depth-pruning space, where structural removals induce inter-layer interactions; its interpretable task-layer profiling guides feasible mask search, making the search both task-aware and trade-off-aware.

3 Method

SliMOO introduces a multi-objective task-aware framework for LLM depth pruning (*cf.* Figure 2). The method first measures how each removable layer or sublayer affects different capability domains, forming a task-layer importance map. This map is then used as a structured prior that guides candidate generation inside a Pareto-aware evolutionary search under an exact sparsity budget. We first define the pruning search space, review the multi-objective optimization preliminaries, and establish our motivation (*cf.* §3.1); we then introduce task-layer importance profiling (*cf.* §3.2) and describe the task-wise objective and TLIP-guided Pareto search procedure (*cf.* §3.3).

3.1 Preliminaries

Pruning units and search space. Consider an LLM with N transformer layers. SliMOO supports two depth-pruning granularities: removing an entire transformer layer, or removing individual

attention/MLP sublayers, inducing search dimensions of $D = N$ and $D = 2N$, respectively. A pruning pattern is represented by a binary mask $\mathbf{x} \in \{0,1\}^D$, where $x_i = 1$ indicates that unit i is removed. In practice, some units may be excluded from the search a priori: for example, the first and last few layers are commonly more sensitive and are rarely safe pruning targets. We encode the remaining removable units with a feasible set $\mathcal{F}$. Under an exact budget of K removed units, SliMOO searches over:

$$\mathcal{X}_K = \{\mathbf{x} \in \{0,1\}^D \mid \mathrm{supp}(\mathbf{x}) \subseteq \mathcal{F},\ \|\mathbf{x}\|_1 = K\}. \tag{1}$$

Multi-objective optimization. For a candidate mask $\mathbf{x}$, let $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_T(\mathbf{x}))$ denote its task-wise pruning losses on T tasks, where all objectives are minimized. Multi-objective optimization seeks masks that are simultaneously good across tasks rather than collapsing heterogeneous capabilities into a single scalar score. A candidate mask $\mathbf{x}$ dominates $\mathbf{y}$ if it is no worse on every objective and strictly better on at least one:

$$\mathbf{x} \prec \mathbf{y} \iff f_t(\mathbf{x}) \le f_t(\mathbf{y})\ \forall t, \quad \exists t' : f_{t'}(\mathbf{x}) < f_{t'}(\mathbf{y}). \tag{2}$$

The corresponding Pareto-optimal set is $\mathcal{P} = \{\mathbf{x} \in \mathcal{X}_K \mid \nexists\, \mathbf{y} \in \mathcal{X}_K :\ \mathbf{y} \prec \mathbf{x}\}$. The evolutionary search iteratively generates offspring candidates from the current population through variation operators, then selects the next generation from the union of parents and offspring. NSGA-II provides a natural environmental selection mechanism for this process: it partitions candidates into non-dominated fronts, retains lower-rank fronts first, and promotes diversity within the last retained front through the crowding distance:

$$\mathrm{CD}(\mathbf{x}) = \sum_{t=1}^{T} \frac{f_t(\mathbf{x}_t^+) - f_t(\mathbf{x}_t^-)}{f_t^{\max} - f_t^{\min}}, \tag{3}$$

where $\mathbf{x}_t^-$ and $\mathbf{x}_t^+$ are the immediate neighbors of candidate $\mathbf{x}$ after sorting the current front by objective t, and $f_t^{\max}/f_t^{\min}$ are the front-wise maximum/minimum values for normalization. Boundary candidates are assigned infinite distance. This rule favors candidates in sparse regions of the same Pareto front, preserving diverse trade-offs instead of collapsing to masks that optimize only one capability domain.

Motivation. A single global pruning order would only be appropriate if all tasks depended on similar layers. Figure 2(a) shows that this is not the case: for example, the MLP unit at layer index 8 is important across all tasks, whereas the MLP unit at layer index 40 is mainly important for code. SliMOO is therefore designed around two complementary requirements: the search should know which units are risky for each task, and selection should preserve masks that realize different cross-domain trade-offs instead of forcing one scalar preference.

3.2 Task-layer Importance Profiling

SliMOO begins with **task-layer importance profiling (TLIP)**, a task-aware probing method for quantifying the contribution of pruning units to different tasks before the Pareto-aware evolutionary search begins. For each removable unit i and task t, TLIP removes only that unit and measures the resulting task degradation:

$$I_{t,i} = f_t(\mathbf{e}_i) - f_t(\mathbf{0}), \tag{4}$$

where $\mathbf{0}$ denotes the dense model, $\mathbf{e}_i$ removes only unit i (f_t is defined below). This singleton probe is cheap relative to evaluating all pruning combinations, but it exposes first-order task sensitivity over depth. Repeating it over all units yields a task-layer importance map $\mathbf{I} \in \mathbb{R}^{T \times D}$. This map plays a dual role. First, it provides an interpretable view of layer contributions across tasks: each row characterizes one task's sensitivity over depth, while each column indicates whether a unit is broadly shared or selectively important to a particular task. Second, it supplies a family of task-layer priors for Pareto-aware search: the task-specific rows $I_{t,:}$ guide exploration toward individual task trade-offs, while a balanced prior is obtained by conservatively aggregating them:

$$\hat{I}_i = \max_t I_{t,i}, \tag{5}$$

This max-aggregated prior protects any unit that is critical to at least one task and is therefore used as the risk-averse guide. During search, SliMOO alternates between $\hat{I}$ and task-specific rows of $\mathbf{I}$: the former steers pruning away from broadly important units, while the latter injects task-focused variation and helps the population explore different Pareto regions.

3.3 Pareto-aware Evolutionary Search

Task-wise pruning objective. Because our goal is to preserve capability across tasks rather than optimize a single proxy, SliMOO evaluates each pruning mask by how closely the pruned model matches the dense model on each task. This avoids requiring task labels for every calibration token and directly measures distributional distortion caused by pruning. For task t, let $c^{(t)}_{n,<j}$ denote the prefix context of the j-th token in calibration example n, and let $m^{(t)}_{n,j} \in \{0,1\}$ be a token-level loss mask. We define the pruning objective as a masked, entropy-normalized forward KL divergence:

$$f_t(\mathbf{x}) = \frac{1}{H^{(t)} + \epsilon} \cdot \frac{1}{Z_t} \sum_{n,j} m^{(t)}_{n,j} \, \mathrm{KL}\big(p_0(\cdot \mid c^{(t)}_{n,<j}) \parallel p_{\mathbf{x}}(\cdot \mid c^{(t)}_{n,<j})\big), \tag{6}$$

where p_0 and $p_{\mathbf{x}}$ are the dense and pruned next-token distributions, $Z_t = \sum_{n,j} m^{(t)}_{n,j}$, and $H^{(t)}$ is the dense-model entropy averaged over the same active positions. Here, $H^{(t)}$ acts as a task-scale normalizer, so the KL values from high- and low-entropy domains are more comparable. The token mask $m^{(t)}$ enables task-specific scoring, such as response-only masking for math and code evaluation. Collecting these task-wise objectives yields the multi-objective optimization problem:

$$\min_{\mathbf{x} \in \mathcal{X}_K} \mathbf{f}(\mathbf{x}) = \big(f_1(\mathbf{x}), \ldots, f_T(\mathbf{x})\big). \tag{7}$$

TLIP-guided Pareto-aware evolutionary search. A vanilla NSGA-II search would treat pruning as a black-box vector optimization problem: it generates masks, evaluates them, and selects the next population by Pareto rank and diversity. SliMOO instead injects task-aware structure into candidate generation through TLIP (*cf.* Figure 2(b)). TLIP is profiled once before search, and its priors are reused for initialization and mutation. Initialization combines importance-guided masks, which sample K distinct units from $\mathcal{F}$ without replacement, biased toward low $\hat{I}$, with random feasible masks for diversity. During evolution, mutation parents are sampled uniformly from the population, crossover uses two distinct uniform parents, and mutations alternate between using $\hat{I}$ and uniformly sampling one task-specific row from $\{I_{t,:}\}_{t=1}^{T}$ as $\tilde{I}$. For a parent with pruned set $S \subseteq \mathcal{F}$, swap mutation samples a kept feasible index $a \in \mathcal{F} \setminus S$ to prune and a pruned index $r \in S$ to restore:

$$P_{\mathrm{prune}}(a) \propto \exp(-\tilde{I}_a/\tau), \qquad P_{\mathrm{restore}}(r) \propto \exp(\tilde{I}_r/\tau), \tag{8}$$

where τ controls exploration versus exploitation: smaller values focus mutation on low-importance units, while larger values flatten sampling. We adapt τ from population hypervolume [27] progress, decaying it after improvement and increasing it after stagnation. In parallel, cardinality-preserving crossover recombines parent pruning sets by inheriting shared removals and filling the remaining budget from their disagreement, so every offspring remains in $\mathcal{X}_K$. Candidate quality is measured by task-wise objectives on calibration sets rather than held-out evaluation benchmarks. NSGA-II then retains non-dominated and diverse trade-offs, and hypervolume tie-breaking prioritizes candidates with larger Pareto contribution for final selection. Algorithm 1 summarizes the full procedure.

4 Experiments

4.1 Experimental setup

Models and pruning settings. We evaluate SliMOO on four main large language models: Llama-3.1-70B, Llama-3.1-405B, Qwen3-32B, and Qwen3-235B-A22B. We study two pruning granularities: whole-layer pruning at 25% sparsity and attention/MLP sublayer pruning at 33% sparsity.

Calibration. SliMOO uses four domain-specific sources: `fineweb_edu` [28] for language modeling, `commonsense_170k` [29] for knowledge reasoning, `GSM8K` (train split) [30] plus `DeepScaleR-Preview` [31] for math, and `python-codes` [32] for code. For math and code, we follow a supervised response-only masking scheme and compute the objective only on solution or code-completion tokens. The multi-domain budget is split evenly across four domains, yielding $65{,}536$ tokens per domain with maximum length $8{,}192$; the math budget is split evenly between `GSM8K` (easy) and `DeepScaleR-Preview` (hard). For the main comparison, calibration-based baselines use the same total budget but the generic source `fineweb_edu`; the fair comparison additionally reruns them on SliMOO's mixed multi-domain pool to isolate calibration-source effects.

Algorithm 1 SliMOO: TLIP-guided Pareto-aware evolutionary search for depth pruning

Input: Dense model $\mathcal{M}$, task calibration sets $\{\mathcal{D}_t\}_{t=1}^{T}$, feasible search space $\mathcal{X}_K$, initial set size M_0, population size P, generations G, offspring budget B, crossover ratio ρ
Output: Selected pruning mask $\mathbf{x}^\star$ and task-layer importance map $\mathbf{I}$
Phase 1: Task-layer importance profiling
for each task $t \in \{1, \ldots, T\}$ **do**
  for each removable unit i **do**
    Compute single-removal effect $I_{t,i} = f_t(\mathbf{e}_i) - f_t(\mathbf{0})$
Form the balanced task-layer prior $\hat{I}$ with $\hat{I}_i \leftarrow \max_t I_{t,i}$
Phase 2: Prior-guided initialization
Generate $\mathcal{C}_0 = \{\mathbf{x}^{(m)}\}_{m=1}^{M_0}$ with importance-guided samples and random feasible masks
Evaluate each $\mathbf{x} \in \mathcal{C}_0$ to obtain its task-wise objective vector $\mathbf{f}(\mathbf{x})$ in Eq. (7)
Select the initial population $\mathcal{P}_0 \leftarrow \text{NSGA-II}(\mathcal{C}_0, P)$
Initialize adaptive temperature τ and offspring sizes $B_{\text{xo}} \leftarrow \lfloor \rho B \rfloor$, $B_{\text{mut}} \leftarrow B - B_{\text{xo}}$
Phase 3: Pareto evolutionary search
for $g = 1$ to G **do**
  Set the parent pool $\mathcal{Q}_g \leftarrow \mathcal{P}_{g-1}$ for uniform parent sampling
  Generate $\mathcal{O}_g^{\text{xo}} = \{\mathbf{x}_{g,b}^{\text{xo}}\}_{b=1}^{B_{\text{xo}}}$
    by inheriting shared pruned units and filling remaining slots from parent disagreements
  Generate $\mathcal{O}_g^{\text{mut}} = \{\mathbf{x}_{g,b}^{\text{mut}}\}_{b=1}^{B_{\text{mut}}}$
    by alternating between $\hat{I}$ and a uniformly sampled task row as $\tilde{I}$, then applying Eq. (8)
  Form candidate pool $\mathcal{C}_g \leftarrow \mathcal{P}_{g-1} \cup \mathcal{O}_g^{\text{xo}} \cup \mathcal{O}_g^{\text{mut}}$
  Evaluate each $\mathbf{x} \in \mathcal{C}_g$ to obtain its task-wise objective vector $\mathbf{f}(\mathbf{x})$ in Eq. (7) ▷ reuse parent scores
  Select $\mathcal{P}_g \leftarrow \text{NSGA-II}(\mathcal{C}_g, P)$
  Adapt τ from population HV progress
Select $\mathbf{x}^\star \leftarrow \text{HV-BEST}(\mathcal{P}_G)$
Return $\mathbf{x}^\star, \mathbf{I}$

We compare against representative depth and structured pruning baselines: ShortGPT [8], LaCo [9], EvoPress [11], BlockPruner [10], FinerCut [12], and LLM-Pruner [33]. LLM-Pruner, a head/neuron pruning method, is included to broaden the comparison beyond pure depth pruning. LaCo merges LLM layers without calibration, which is evaluated in its original calibration-free setting.

Evaluation tasks. We evaluate downstream performance on four profiled capability domains:

1. *Language modeling:* WikiText-2 [34], PTB [35], and C4 [36] perplexity (lower is better).
2. *Math:* GSM8K [30] (5-shot exact match) and Hendrycks MATH [37] (4-shot exact match).
3. *Code:* HumanEval [38] (0-shot pass@1) and MBPP [39] (3-shot pass@1).
4. *Knowledge reasoning:* MMLU [40] (5-shot accuracy), plus ARC-Challenge, ARC-Easy [41], Social-IQA [42], HellaSwag [43], PIQA [44], and WinoGrande [45] (0-shot accuracy).

We additionally include five auxiliary benchmarks for out-of-domain evaluation: RULER [46] (NIAH-8K) for long-context retrieval, TruthfulQA [47] (MC2) for truthfulness, CrowS-Pairs [48] for social bias, ToxiGen [49] for toxicity and implicit hate speech, and MedMCQA [50] for medical question answering. These benchmarks are excluded from TLIP and evolutionary search.

Implementation details. Our implementation is built with `PyTorch` [51] and uses Hugging Face `Transformers` [52] and `Datasets` [53] to manage models and data. TLIP uses $8{,}192$ profiling tokens per task. For all models, the first two and last two layers are treated as pruning-sensitive and excluded from the search space. SliMOO search runs use 30 generations, population size 4, 64 offspring per generation, 128 initial candidates, and crossover rate 0.3. For initialization, $\tau = 1.0$: half of the candidates are importance-guided samples with probabilities proportional to $\exp(-\hat{I}_i/\tau)$, and half are random feasible masks for diversity; when generating offspring, τ decays by 0.9 on population HV improvement and grows by 1.5 after 3 stagnant generations, clipped to $[0.1, 5.0]$.

4.2 Main results

Post-pruning performance. Table 1 shows that preserving a single domain is not sufficient for reliable depth pruning. Under the 25% whole-layer sparsity setting, SliMOO consistently improves the strongest multi-task trade-offs across both scales. On Llama-3.1-70B, ShortGPT remains reasonably

Table 1: Post-pruning performance on the Llama family. We report domain averages: lower is better for language modeling and higher is better for math, code, and knowledge reasoning. **Bold** indicates the best result, and underlines indicate the runner-up.

Model	Sparsity	Method	Language Avg. ↓	Math Avg. ↑ %	Code Avg. ↑ %	Knowledge Avg. ↑ %
Llama-3.1-70B	0%	-	5.73	52.48	58.32	72.25
	25% Layer	ShortGPT	11.35	12.85	15.45	65.93
		LaCo	11.20	16.00	20.39	66.22
		LLM-Pruner	**8.74**	32.44	26.78	66.78
		EvoPress	9.88	24.33	35.03	64.63
		SliMOO	9.89	**34.40**	**39.14**	**66.90**
	33% Sublayer	ShortGPT	35.22	5.20	2.75	63.38
		EvoPress	12.02	10.33	15.11	62.02
		BlockPruner	10.99	24.22	18.89	63.96
		FinerCut	12.85	13.10	7.87	64.73
		SliMOO	**10.45**	**27.27**	**30.78**	**66.29**
Llama-3.1-405B	0%	-	4.68	60.41	62.94	75.01
	25% Layer	ShortGPT	8.10	28.68	48.73	70.58
		LaCo	8.71	40.69	49.26	70.59
		EvoPress	**7.45**	44.95	46.90	70.22
		SliMOO	7.59	**49.83**	**51.99**	**71.03**
	33% Sublayer	ShortGPT	22.41	1.56	0.00	33.02
		EvoPress	9.26	29.32	41.25	67.10
		BlockPruner	7.53	42.49	40.13	69.25
		FinerCut	9.97	45.51	43.81	71.13
		SliMOO	**7.27**	**54.57**	**59.95**	**71.71**

competitive on the knowledge reasoning-domain average (65.93%), but collapses to 12.85% and 15.45% on the math- and code-domain averages; SliMOO raises these to 34.40% and 39.14% while also achieving the best knowledge reasoning-domain average (66.90%). Relative to the calibration-free LaCo and the evolutionary search baseline EvoPress, SliMOO delivers the largest gains on math and code. We also report LLM-Pruner as a structural pruning reference: although it achieves the best language-modeling average on Llama-3.1-70B at 25% sparsity, it remains weaker than SliMOO on math, code, and knowledge reasoning. Its 405B results are omitted due to memory constraints.

At 33% sparsity, where pruning operates at the attention/MLP sublayer level, the problem becomes harder and SliMOO's advantage becomes clearer. On Llama-3.1-70B, it improves the math/code-domain averages from $10.33\%/15.11\%$ under EvoPress to $27.27\%/30.78\%$. On Llama-3.1-405B, it surpasses FinerCut by $+9.06$ points on math and $+16.14$ points on code, while still achieving the best language modeling average and remaining competitive on knowledge reasoning. The detailed results for both the Llama and Qwen families are reported in Appendix A.2.

Search dynamics. Figure 3 visualizes the evolution of SliMOO's search objectives across generations. All task-wise objectives drop sharply in the first few generations, showing that TLIP-guided initialization and mutation quickly move the population toward promising pruning regions. The curves then improve more gradually, reflecting the role of Pareto-aware evolution in refining trade-offs rather than simply optimizing the easiest domain. Importantly, no objective shows persistent deterioration or collapse, suggesting that SliMOO maintains balanced progress across language, math, code, and knowledge reasoning throughout search.

4.3 Generalization to Unseen Tasks

We evaluate the existing Llama-3.1-70B pruning mask on five out-of-domain benchmarks without additional profiling, search, or fine-tuning, as shown in Table 2.

SliMOO achieves the highest TruthfulQA, ToxiGen, and MedMCQA scores among the evaluated pruned models. At 25% layer sparsity, it exceeds the strongest mixed-calibration baseline by 1.96 and 2.66 points on TruthfulQA and MedMCQA, respectively. At 33% sublayer sparsity, the corresponding gains are 0.70 and 0.12 points, while EvoPress-mix retains a higher in-domain math score (28.29 vs. 27.27). RULER-NIAH is nearly saturated across methods, and SliMOO remains closest to the dense model's CrowS-Pairs behavior. These results suggest that balancing heterogeneous in-domain

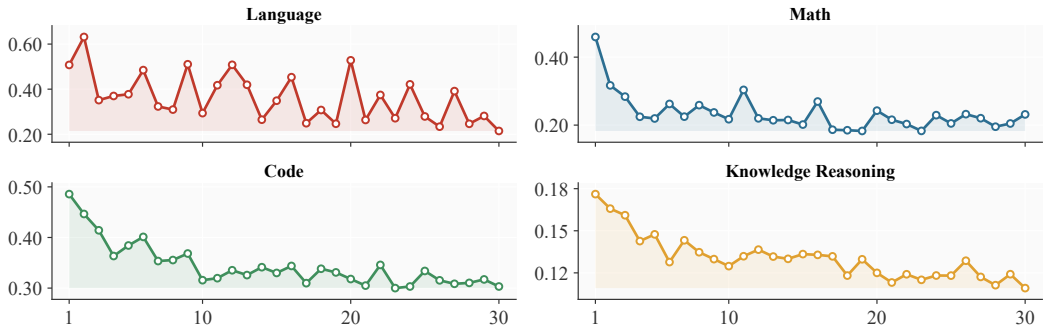


Figure 3: Search dynamics of SliMOO across four capability domains. The horizontal axis denotes the generation index, and the vertical axis reports the corresponding domain-specific search objectives used during optimization; lower values indicate smaller distributional deviation from the dense model.

Table 2: Performance on the four profiled capability domains and five out-of-domain benchmarks for Llama-3.1-70B. `-mix` denotes calibration using the same mixed-domain samples and total token budget as SliMOO. All out-of-domain evaluations use the existing pruning mask without additional profiling, search, or fine-tuning.

Sparsity	Method	Profiled capability domains				Out-of-domain benchmarks				
		Lang. ↓	Math ↑	Code ↑	Know. ↑	RULER NIAH-8K ↑	TruthfulQA MC2 ↑	CrowS Pairs†	ToxiGen ↑	MedMCQA ↑
0%	Dense	5.73	52.48	58.32	72.25	100.0	51.14	68.63	46.49	69.28
25% Layer	ShortGPT	11.35	12.85	15.45	65.93	99.8	44.80	63.80	42.55	56.30
	ShortGPT-mix	11.77	13.17	19.48	65.43	99.9	45.53	64.70	43.72	60.00
	EvoPress	**9.88**	24.33	35.03	64.63	100.0	42.60	65.24	43.19	54.65
	EvoPress-mix	11.35	27.89	35.62	66.12	100.0	46.27	65.06	44.36	58.04
	SliMOO	9.89	**34.40**	**39.14**	**66.90**	100.0	**48.23**	**66.13**	**44.68**	**62.66**
33% Sublayer	EvoPress	12.02	10.33	15.11	62.02	100.0	44.39	63.63	41.38	53.35
	EvoPress-mix	10.92	**28.29**	24.04	64.46	100.0	46.14	64.28	42.45	56.06
	FinerCut	12.85	13.10	7.87	64.73	100.0	43.45	62.85	38.29	52.55
	FinerCut-mix	13.41	18.18	16.32	62.65	99.9	44.92	63.51	40.32	53.26
	SliMOO	**10.45**	27.27	**30.78**	**66.29**	100.0	**46.84**	**65.41**	**42.66**	**56.18**

† CrowS-Pairs reports stereotypical preference. Closeness to the dense-model score indicates behavioral retention.

objectives steers pruning toward layers that are broadly redundant across capabilities, helping preserve performance on tasks outside the search objectives.

4.4 Multilingual Domain Transfer

To test whether SliMOO remains useful beyond the capability domains used in the main experiments, we construct a multilingual pruning scenario on Qwen3-32B, a choice with native multilingual support. The calibration budget matches the main setting but is redistributed across `FineWeb-Edu` English and `FineWeb-2` Spanish, French, and German subsets [54]. Evaluation uses per-token perplexity on C4 multilingual validation splits for English, Spanish, French, and German, with 64 windows of 4096 tokens per language. As shown in Table 3, all pruning methods increase perplexity, but SliMOO consistently yields the lowest perplexity among pruned models across the four languages. This suggests that SliMOO is not tied to a fixed domain taxonomy: language/math/code/knowledge and multilingual partitions are examples of how its task-aware multi-objective formulation can be instantiated.

Table 3: Multilingual C4 perplexity on Qwen3-32B after 25% layer pruning. Lower is better, and **bold** marks the best pruned model.

Method	EN ↓	ES ↓	FR ↓	DE ↓
Dense	12.19	9.81	6.56	6.94
ShortGPT	17.75	10.56	8.75	9.31
EvoPress	16.01	10.18	8.22	8.89
SliMOO	**14.69**	**9.81**	**7.62**	**8.62**

Table 4: Fair calibration comparison on Llama-3.1-70B. All baselines are rerun with the same mixed multi-domain calibration data and total token budget as SliMOO.

Sp.	Method	Lang. ↓	Math ↑	Code ↑	Know. ↑
25% Layer	ShortGPT	11.77	13.17	19.48	65.43
	LLM-Pruner	**9.42**	32.44	32.30	66.15
	EvoPress	11.35	27.89	35.62	66.12
	SliMOO	9.89	**34.40**	**39.14**	**66.90**
33% Sublayer	ShortGPT	20.53	8.71	3.85	64.18
	EvoPress	10.92	**28.29**	24.04	64.46
	BlockPruner	11.38	24.16	21.43	65.40
	FinerCut	13.41	18.18	16.32	62.65
	SliMOO	**10.45**	27.27	**30.78**	**66.29**

Table 5: Calibration budget and allocation ablation for SliMOO on Llama-3.1-70B at 25% layer sparsity. In the dominant (dom.) setting, one domain receives half of the total budget and the rest is evenly split.

Calib.	Lang. ↓	Math ↑	Code ↑	Know. ↑
Default (64K)	9.89	34.40	39.14	66.90
32K/domain	9.89	34.40	39.14	66.90
16K/domain	10.13	30.25	**40.19**	66.82
8K/domain	10.39	29.04	37.74	66.40
Lang.-dom.	**9.81**	29.27	34.44	65.47
Math-dom.	9.89	**35.16**	28.46	65.85
Code-dom.	10.15	18.29	39.26	66.15
Know.-dom.	9.97	20.04	25.07	**67.69**

Table 6: Ablation of search objectives and Pareto selection on Llama-3.1-70B at 25% layer sparsity.

Search variant	Lang. ↓	Math ↑	Code ↑	Know. ↑
Language Evolution	**9.88**	24.33	35.03	64.63
Math Evolution	10.75	**36.11**	34.89	66.44
Code Evolution	9.89	30.93	38.59	65.66
Knowledge Evolution	11.35	25.46	27.20	**67.45**
Mixed Evolution	11.35	27.89	35.62	66.12
Mean Evolution	9.97	29.14	33.10	66.62
Random Pareto Evolution	9.96	29.06	33.11	66.27
HV Evolution	9.91	30.77	34.73	66.58
NSGA-II	9.89	32.67	36.02	66.71
SliMOO	9.89	34.40	**39.14**	66.90

Table 7: Ablation of key SliMOO hyperparameters on Llama-3.1-70B at 25% layer sparsity.

Configuration	Lang. ↓	Math ↑	Code ↑	Know. ↑
SliMOO	9.89	34.40	39.14	66.90
20 Generations	12.10	29.49	35.77	66.36
40 Generations	10.03	35.44	37.92	66.57
32 Offspring	10.25	32.56	35.79	66.18
128 Offspring	10.07	**35.64**	40.21	65.77
Crossover Rate 0.1	10.18	33.20	37.71	**67.03**
Crossover Rate 0.5	9.89	34.40	39.14	66.90
Population Size 6	**9.78**	33.08	**40.61**	66.79
Fixed temp. $\tau = 1$	10.44	32.61	37.27	66.37

4.5 Ablation Study

Fair calibration comparison. Table 4 reports a controlled comparison where all calibration-based baselines are rerun with the same mixed multi-domain calibration data and total token budget as SliMOO. The same conclusion remains: SliMOO's gains cannot be attributed to calibration allocation alone. At 25% layer sparsity, SliMOO improves the math/code averages from 27.89%/35.62% under EvoPress to 34.40%/39.14% and achieves the best knowledge average (66.90%). At 33% sublayer sparsity, it attains the best language/code/knowledge averages (10.45/30.78%/66.29%) while remaining competitive on math (27.27% versus 28.29% under EvoPress), yielding the strongest overall trade-off. This further supports that SliMOO's advantage comes from its task-wise objective and Pareto-guided search rather than from task-aware calibration allocation alone.

Calibration budget and allocation. Table 5 shows that SliMOO is robust to calibration size. Halving the budget from 64K to 32K tokens per domain recovers the same pruning mask and identical post-pruning scores (9.89/34.40%/39.14%/66.90% on language/math/code/knowledge), indicating that the task-layer prior is already stable at moderate calibration sizes. Further reducing the budget to 16K or 8K introduces some degradation, but the resulting models remain competitive rather than collapsing, suggesting that SliMOO does not rely on a large calibration set. The allocation study shows a complementary effect: domain-dominant calibration can steer the selected mask toward that domain, with math-dominant calibration giving the best math average (35.16%) and knowledge-dominant calibration giving the best knowledge average (67.69%) while trading off other domains. Adjusting domain proportions is therefore a practical way to reflect user preferences, while equal allocation remains the default for balanced multi-domain pruning.

Search objective and Pareto selection. Table 6 first shows that single-objective evolutionary search improves its target domain but fails to preserve the full capability profile. The multi-objective variants are more balanced but still incomplete: *Mixed Evolution* applies evolutionary search on mixed calibration data, *Mean Evolution* computes each task objective separately and averages them into one scalar, *Random Pareto Evolution* first filters the non-dominated candidates and then samples from them uniformly, and *HV Evolution* keeps candidates with larger hypervolume contribution. These alternatives improve over single-domain search but remain weaker than NSGA-II, which explicitly

retains non-dominated and diverse candidates. SliMOO further adds TLIP profiling to guide candidate generation toward task-relevant regions, yielding the strongest overall trade-off across domains.

Ablation on SliMOO search hyperparameters. Table 7 shows that SliMOO is largely robust: changing generations, offspring, crossover, or population size does not cause collapse and still yields competitive trade-offs. Reducing the search to 20 generations weakens all domains, while 40 generations improves math but does not dominate the default balance. Increasing the offspring budget to 128 further improves math and code but hurts knowledge reasoning, indicating a different Pareto point rather than a uniformly better solution. Crossover is stable: rates 0.3 and 0.5 recover the same mask, while a smaller rate slightly favors knowledge. A larger population improves language and code but sacrifices math. Finally, fixed τ degrades the trade-off, suggesting that dynamic exploration helps maintain a balanced Pareto search.

4.6 Efficiency Analysis

Efficiency of SliMOO framework. All experiments are conducted on AMD Instinct MI300X GPUs. On Llama-3.1-70B with 2 GPUs, TLIP profiling takes 12.0 minutes and SliMOO search takes 20.8 minutes per generation, close to EvoPress (18.8 minutes $\times$ 30 generations) while producing stronger masks. ShortGPT is fast because it only requires several forward passes, but its post-pruning quality is substantially weaker; FinerCut is much slower, taking 39.2 hours to finish. For Llama-3.1-405B, SliMOO runs on 8 GPUs with a 933 GB memory budget; profiling takes 46.4 minutes and search takes 52.1 minutes per generation. Thus, SliMOO's one-time offline cost remains acceptable given its best post-pruning trade-offs.

Table 8: Inference efficiency of pruned Llama-3.1-70B with vLLM at batch size 8.

Sparsity	Arch	IO Tok.	TTFT (s) ↓	Decode TPS ↑
0%	80 Attn	4096	3.39	482.9
	80 FFN	8192	4.47	413.6
25%	60 Attn	4096	2.54	623.9
	60 FFN	8192	3.35	539.5
33%	46 Attn	4096	2.37	672.2
	64 FFN	8192	3.07	593.0

Efficiency of post-pruning LLMs. Table 8 reports time to first token (TTFT) and decode throughput (Decode TPS), showing that SliMOO converts structural pruning into practical serving gains. SliMOO prunes the full-depth Llama-3.1-70B model into layer- or sublayer-sparse variants, enabling more efficient serving with standard vLLM. At 4096 IO tokens (equal input and output tokens), 25% layer pruning gives a $1.33\times$ TTFT speedup and $1.29\times$ Decode TPS improvement, while 33% sublayer pruning increases these gains to $1.43\times$ and $1.39\times$. Similar gains hold at 8192 IO tokens, confirming faster inference without specialized sparse kernels.

5 Limitations

SliMOO achieves strong and interpretable cross-domain trade-offs, yet it still has limitations. Concretely, the current instantiation of SliMOO uses a predefined set of capability domains and calibration corpora, which locally aligns it with deployment settings whose target domains can be reasonably specified, while broader open-ended, in-the-wild task mixtures merit further study. SliMOO also incurs a one-time cost for TLIP profiling and evolutionary search, so reducing this overhead would improve practicality. For sublayer pruning, our comparisons follow the same structural-unit budget across methods, while attention and MLP units can differ in parameter and FLOP counts; nevertheless, vLLM measurements show that higher-sparsity sublayer pruning can yield noticeably stronger serving speedups than low-sparsity layer pruning even when their parameter counts are relatively close.

6 Conclusion

This paper presents **SliMOO**, an interpretable multi-objective framework for LLM depth pruning. By profiling task-wise layer importance and implementing Pareto-aware evolutionary search, SliMOO avoids the limitations of a single global pruning order and instead identifies pruning masks with stronger cross-domain trade-offs under fixed sparsity budgets. Across the Llama-3.1 and Qwen3 families, SliMOO significantly improves post-pruning performance in the domains of language modeling, math, code, and knowledge reasoning. These results suggest that task-aware multi-objective optimization is a promising direction for reliable and interpretable LLM depth pruning.

Acknowledgments and Disclosure of Funding

This work received no external funding. The authors declare no additional competing interests beyond the affiliations disclosed in the paper.

References

[1] DeepSeek-AI, "DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning," *arXiv preprint arXiv:2501.12948*, 2025. [Online]. Available: https://arxiv.org/abs/2501.12948

[2] Llama Team, "The Llama 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024. [Online]. Available: https://arxiv.org/abs/2407.21783

[3] Qwen Team, "Qwen3 Technical Report," *arXiv preprint arXiv:2505.09388*, 2025. [Online]. Available: https://arxiv.org/abs/2505.09388

[4] G. Li, Y. Xu, Z. Li, J. Liu, X. Yin, D. Li, and E. Barsoum, "Týr-the-Pruner: Structural pruning LLMs via global sparsity distribution optimization," in *Proc. Conf. Neural Information Processing Systems (NeurIPS)*, 2025. [Online]. Available: https://openreview.net/forum?id=rAuRLePL2R

[5] M. Sun, Z. Liu, A. Bair, and J. Z. Kolter, "A simple and effective pruning approach for large language models," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2024. [Online]. Available: https://openreview.net/forum?id=PxoFut3dWW

[6] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, "GPTQ: Accurate post-training quantization for generative pre-trained transformers," *arXiv preprint arXiv:2210.17323*, 2022. [Online]. Available: https://arxiv.org/abs/2210.17323

[7] D. Joo, H. Hosseini, R. Hadidi, and B. Asgari, "Coruscant: Co-designing GPU kernel and sparse tensor core to advocate unstructured sparsity in efficient LLM inference," in *Proc. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, 2025, pp. 232–245. [Online]. Available: https://doi.org/10.1145/3725843.3756065

[8] X. Men, M. Xu, Q. Zhang, Q. Yuan, B. Wang, H. Lin, Y. Lu, X. Han, and W. Chen, "ShortGPT: Layers in large language models are more redundant than you expect," in *Findings of the Association for Computational Linguistics: ACL*, 2025, pp. 20 192–20 204. [Online]. Available: https://aclanthology.org/2025.findings-acl.1035/

[9] Y. Yang, Z. Cao, and H. Zhao, "LaCo: Large language model pruning via layer collapse," in *Findings of the Association for Computational Linguistics: EMNLP*, 2024, pp. 6401–6417. [Online]. Available: https://doi.org/10.18653/v1/2024.findings-emnlp.372

[10] L. Zhong, F. Wan, R. Chen, X. Quan, and L. Li, "BlockPruner: Fine-grained pruning for large language models," in *Findings of the Association for Computational Linguistics: ACL*, 2025, pp. 5065–5080. [Online]. Available: https://aclanthology.org/2025.findings-acl.262/

[11] O. Sieberling, D. Kuznedelev, E. Kurtic, and D. Alistarh, "EvoPress: Accurate dynamic model compression via evolutionary search," in *Proc. Int. Conf. Machine Learning (ICML)*, 2025. [Online]. Available: https://proceedings.mlr.press/v267/sieberling25a.html

[12] Y. Zhang, Y. Li, X. Wang, Q. Shen, B. Plank, B. Bischl, M. Rezaei, and K. Kawaguchi, "FinerCut: Finer-grained interpretable layer pruning for large language models," in *Workshop on Machine Learning and Compression, NeurIPS*, 2024. [Online]. Available: https://openreview.net/forum?id=jrSWzgno4W

[13] T. Schuster, A. Fisch, J. Gupta, M. Dehghani, D. Bahri, V. Tran, Y. Tay, and D. Metzler, "Confident adaptive language modeling," in *Proc. Conf. Neural Information Processing Systems (NeurIPS)*, 2022. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/hash/6fac9e316a4ae75ea244ddcef1982c71-Abstract-Conference.html

[14] M. Elhoushi, A. Shrivastava, D. Liskovich, B. Hosmer, B. Wasti, L. Lai, A. Mahmoud, B. Acun, S. Agarwal, A. Roman, A. A. Aly, B. Chen, and C.-J. Wu, "LayerSkip: Enabling early exit inference and self-speculative decoding," in *Proc. Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024, pp. 12 622–12 642. [Online]. Available: https://doi.org/10.18653/v1/2024.acl-long.681

[15] D. Raposo, S. Ritter, B. A. Richards, T. P. Lillicrap, P. C. Humphreys, and A. Santoro, "Mixture-of-depths: Dynamically allocating compute in transformer-based language models," *arXiv preprint arXiv:2404.02258*, 2024. [Online]. Available: https://arxiv.org/abs/2404.02258

[16] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. E. Gonzalez, H. Zhang, and I. Stoica, "Efficient memory management for large language model serving with PagedAttention," in *Proc. ACM SIGOPS Symp. Operating Systems Principles (SOSP)*, 2023.

[17] L. Zheng, L. Yin, Z. Xie, C. Sun, J. Huang, C. H. Yu, S. Cao, C. Kozyrakis, I. Stoica, J. E. Gonzalez, C. W. Barrett, and Y. Sheng, "SGLang: Efficient execution of structured language model programs," in *Proc. Conf. Neural Information Processing Systems (NeurIPS)*, 2024. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2024/hash/724be4472168f31ba1c9ac630f15dec8-Abstract-Conference.html

[18] C. M. Fonseca and P. J. Fleming, "Genetic algorithms for multiobjective optimization: Formulation, discussion, and generalization," in *Proc. Int. Conf. Genetic Algorithms*, 1993, pp. 416–423.

[19] N. Srinivas and K. Deb, "Multiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary Computation*, vol. 2, no. 3, pp. 221–248, 1994. [Online]. Available: https://doi.org/10.1162/evco.1994.2.3.221

[20] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, 2001.

[21] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002. [Online]. Available: https://doi.org/10.1109/4235.996017

[22] T. Elsken, J. H. Metzen, and F. Hutter, "Efficient multi-objective neural architecture search via lamarckian evolution," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2019. [Online]. Available: https://openreview.net/forum?id=ByME42AqK7

[23] W. Chen, H. Yang, Y. Gou, H. Shi, E.-L. Hu, Z. Li, and J. Kwok, "Multi-objective one-shot pruning for large language models," in *Proc. Conf. Neural Information Processing Systems (NeurIPS)*, 2025. [Online]. Available: https://openreview.net/forum?id=aNpj43Uh35

[24] I. Tenney, D. Das, and E. Pavlick, "BERT rediscovers the classical NLP pipeline," in *Proc. Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019, pp. 4593–4601. [Online]. Available: https://doi.org/10.18653/v1/P19-1452

[25] Y. Zhang, Y. Dong, and K. Kawaguchi, "Investigating layer importance in large language models," *arXiv preprint arXiv:2409.14381*, 2024. [Online]. Available: https://arxiv.org/abs/2409.14381

[26] Y. Leng and D. Xiong, "Towards understanding multi-task learning (generalization) of LLMs via detecting and exploring task-specific neurons," in *Proc. Int. Conf. Computational Linguistics (COLING)*, 2025, pp. 2969–2987. [Online]. Available: https://aclanthology.org/2025.coling-main.200/

[27] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999. [Online]. Available: https://doi.org/10.1109/4235.797969

[28] G. Penedo, H. Kydlicek, L. B. Allal, A. Lozhkov, M. Mitchell, C. A. Raffel, L. von Werra, and T. Wolf, "The FineWeb datasets: Decanting the web for the finest text data at scale," in *Proc. Conf. Neural Information Processing Systems (NeurIPS)*, 2024. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2024/hash/370df50ccfdf8bde18f8f9c2d9151bda-Abstract-Datasets_and_Benchmarks_Track.html

[29] S.-Y. Liu, C.-Y. Wang, H. Yin, P. Molchanov, Y.-C. F. Wang, K.-T. Cheng, and M.-H. Chen, "DoRA: Weight-decomposed low-rank adaptation," in *Proc. Int. Conf. Machine Learning (ICML)*, 2024, pp. 32 100–32 121. [Online]. Available: https://proceedings.mlr.press/v235/liu24bn.html

[30] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman, "Training verifiers to solve math word problems," *arXiv preprint arXiv:2110.14168*, 2021. [Online]. Available: https://arxiv.org/abs/2110.14168

[31] math-dataset, "DeepScaleR-Preview-Dataset," Hugging Face dataset card, 2025. [Online]. Available: https://huggingface.co/datasets/math-dataset/DeepScaleR-Preview-Dataset

[32] flytech, "python-codes-25k," Hugging Face dataset card, 2024. [Online]. Available: https://huggingface.co/datasets/flytech/python-codes-25k

[33] X. Ma, G. Fang, and X. Wang, "LLM-Pruner: On the structural pruning of large language models," in *Proc. Conf. Neural Information Processing Systems (NeurIPS)*, 2023. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/hash/44956951349095f74492a5471128a7e0-Abstract-Conference.html

[34] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," arXiv preprint arXiv:1609.07843, 2016. [Online]. Available: https://arxiv.org/abs/1609.07843

[35] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz, "Building a large annotated corpus of english: The penn treebank," *Computational Linguistics*, vol. 19, no. 2, pp. 313–330, 1993. [Online]. Available: https://aclanthology.org/J93-2004/

[36] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," arXiv preprint arXiv:1910.10683, 2023. [Online]. Available: https://arxiv.org/abs/1910.10683

[37] D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt, "Measuring mathematical problem solving with the MATH dataset," in *Proc. Conf. Neural Information Processing Systems (NeurIPS)*, 2021.

[38] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, and W. Zaremba, "Evaluating large language models trained on code," *arXiv preprint arXiv:2107.03374*, 2021. [Online]. Available: https://arxiv.org/abs/2107.03374

[39] J. Austin, A. Odena, M. I. Nye, M. Bosma, H. Michalewski, D. Dohan, E. Jiang, C. J. Cai, M. Terry, Q. V. Le, and C. Sutton, "Program synthesis with large language models," *arXiv preprint arXiv:2108.07732*, 2021. [Online]. Available: https://arxiv.org/abs/2108.07732

[40] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt, "Measuring massive multitask language understanding," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2021.

[41] P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord, "Think you have solved question answering? try ARC, the AI2 reasoning challenge," *arXiv preprint arXiv:1803.05457*, 2018. [Online]. Available: https://arxiv.org/abs/1803.05457

[42] M. Sap, H. Rashkin, D. Chen, R. L. Bras, and Y. Choi, "Social IQa: Commonsense reasoning about social interactions," in *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP)*, 2019. [Online]. Available: https://aclanthology.org/D19-1454/

[43] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi, "HellaSwag: Can a machine really finish your sentence?" in *Proc. Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019.

[44] Y. Bisk, R. Zellers, R. L. Bras, J. Gao, and Y. Choi, "PIQA: Reasoning about physical commonsense in natural language," in *Proc. AAAI Conf. Artificial Intelligence*, 2020.

[45] K. Sakaguchi, R. L. Bras, C. Bhagavatula, and Y. Choi, "WinoGrande: An adversarial winograd schema challenge at scale," in *Proc. AAAI Conf. Artificial Intelligence*, 2020.

[46] C.-P. Hsieh, S. Sun, S. Kriman, S. Acharya, D. Rekesh, F. Jia, and B. Ginsburg, "RULER: What's the real context size of your long-context language models?" in *First Conference on Language Modeling*, 2024. [Online]. Available: https://openreview.net/forum?id=kIoBbc76Sy

[47] S. Lin, J. Hilton, and O. Evans, "TruthfulQA: Measuring how models mimic human falsehoods," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, S. Muresan, P. Nakov, and A. Villavicencio, Eds. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 3214–3252. [Online]. Available: https://aclanthology.org/2022.acl-long.229/

[48] N. Nangia, C. Vania, R. Bhalerao, and S. R. Bowman, "CrowS-pairs: A challenge dataset for measuring social biases in masked language models," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds. Online: Association for Computational Linguistics, Nov. 2020, pp. 1953–1967. [Online]. Available: https://aclanthology.org/2020.emnlp-main.154/

[49] T. Hartvigsen, S. Gabriel, H. Palangi, M. Sap, D. Ray, and E. Kamar, "ToxiGen: A large-scale machine-generated dataset for adversarial and implicit hate speech detection," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, S. Muresan, P. Nakov, and A. Villavicencio, Eds. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 3309–3326. [Online]. Available: https://aclanthology.org/2022.acl-long.234/

[50] A. Pal, L. K. Umapathi, and M. Sankarasubbu, "Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering," in *Conference on health, inference, and learning*. PMLR, 2022, pp. 248–260.

[51] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Koepf, E. Z. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Conf. Neural Information Processing Systems (NeurIPS)*, 2019, pp. 8024–8035. [Online]. Available: https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html

[52] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Transformers: State-of-the-art natural language processing," in *Proc. Conf. Empirical Methods in Natural Language Processing: System Demonstrations (EMNLP)*, 2020, pp. 38–45. [Online]. Available: https://doi.org/10.18653/v1/2020.emnlp-demos.6

[53] Q. Lhoest, A. V. del Moral, Y. Jernite, A. Thakur, P. von Platen, S. Patil, J. Chaumond, M. Drame, J. Plu, L. Tunstall, J. Davison, M. Sasko, G. Chhablani, B. Malik, S. Brandeis, T. L. Scao, V. Sanh, C. Xu, N. Patry, A. McMillan-Major, P. Schmid, S. Gugger, C. Delangue, T. Matussiere, L. Debut, S. Bekman, P. Cistac, T. Goehringer, V. Mustar, F. Lagunas, A. M. Rush, and T. Wolf, "Datasets: A community library for natural language processing," in *Proc. Conf. Empirical Methods in Natural Language Processing: System Demonstrations (EMNLP)*, 2021, pp. 175–184. [Online]. Available: https://doi.org/10.18653/v1/2021.emnlp-demo.21

[54] G. Penedo, H. Kydlicek, V. Sabolcec, B. Messmer, N. Foroutan, A. H. Kargaran, C. Raffel, M. Jaggi, L. von Werra, and T. Wolf, "FineWeb2: One pipeline to scale them all – adapting pre-training data processing to every language," arXiv preprint arXiv:2506.20920, 2025. [Online]. Available: https://arxiv.org/abs/2506.20920

[55] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2022. [Online]. Available: https://openreview.net/forum?id=nZeVKeeFYf9

[56] N. Lambert, J. Morrison, V. Pyatkin, S. Huang, H. Ivison, F. Brahman, L. J. V. Miranda, A. Liu, N. Dziri, S. Lyu *et al.*, "Tulu 3: Pushing frontiers in open language model post-training," *arXiv preprint arXiv:2411.15124*, 2024. [Online]. Available: https://arxiv.org/abs/2411.15124

[57] A. Lozhkov, L. B. Allal, L. von Werra, and T. Wolf, "FineWeb-Edu: The finest collection of educational content," Hugging Face dataset card, 2024. [Online]. Available: https://huggingface.co/datasets/HuggingFaceFW/fineweb-edu

A Appendix

A.1 SliMOO Search Details and Derivation

The main text defines the feasible mask set $\mathcal{X}_K$, the task-wise KL objectives $\mathbf{f}(\mathbf{x})$, and the TLIP-guided search procedure. This appendix gives the derivation behind these design choices. The central difficulty is that depth pruning is a fixed-budget combinatorial problem with task-dependent and non-additive effects: removing one layer can be harmless for language modeling but harmful for math, and the effect of removing several layers is not exactly the sum of their isolated effects. Let t index a calibration task, $i \in \{1, \dots, D\}$ index a removable unit, and $x_i \in \{0, 1\}$ indicate whether unit i is pruned. If layer interactions were ignored, the task loss of a mask could be approximated by the singleton expansion:

$$\tilde{f}_t(\mathbf{x}) = \sum_{i=1}^{D} x_i \, I_{t,i}, \tag{9}$$

where $\tilde{f}_t(\mathbf{x})$ is the additive approximation to the true task loss $f_t(\mathbf{x})$, and $I_{t,i} = f_t(\mathbf{e}_i) - f_t(\mathbf{0})$ is the TLIP score obtained by pruning only unit i with the singleton mask $\mathbf{e}_i$. This additive view explains why TLIP is informative: it estimates the first-order discrete sensitivity of each task to each removable unit. It also explains why TLIP alone is insufficient as a pruning rule. Since transformer layers interact, the best K units to remove cannot be determined reliably by sorting singleton scores alone. SliMOO therefore uses TLIP as a search prior rather than as the final objective: the prior narrows exploration toward plausible masks, while the actual candidate quality is still measured by the full vector objective $\mathbf{f}(\mathbf{x})$.

The conservative prior $\hat{I}_i = \max_t I_{t,i}$ follows from the multi-objective nature of the problem. A unit with a large score for any task is potentially capability-critical, even if it appears unimportant for other tasks. Using the maximum therefore protects task-specific layers that would be washed out by averaging. At the same time, SliMOO also samples task-specific rows $I_{t,:}$ during mutation, because a purely conservative prior would over-protect shared high-sensitivity regions and could under-explore domain-specific Pareto trade-offs. In this sense, the balanced prior controls risk, while task-specific priors maintain search diversity.

Feasibility is preserved by viewing masks in $\mathcal{X}_K$ as K-subsets of the feasible unit set $\mathcal{F}$. The most local valid move is a swap: remove one currently kept unit from $\mathcal{F}$ and restore one currently pruned unit. This changes the mask by Hamming distance two but keeps $\|\mathbf{x}\|_1 = K$, so every mutated candidate remains at the target sparsity. TLIP enters this move only through the proposal distribution: low-prior kept units are more likely to be newly pruned, and high-prior removed units are more likely to be restored. Cardinality-preserving crossover serves the same purpose at the population level. Given two parent pruning sets S_1 and S_2 with $|S_1| = |S_2| = K$, the offspring first inherits their shared pruned units $S_1 \cap S_2$, and then fills the remaining slots by sampling from the symmetric difference $(S_1 \cup S_2) \setminus (S_1 \cap S_2)$ until its size reaches K. This recombines useful pruning patterns from both parents while guaranteeing that the child remains in $\mathcal{X}_K$.

After candidate generation, selection is performed in objective space. Let $\mathcal{C}$ denote the candidate pool formed by the current population and newly generated offspring. NSGA-II constructs non-dominated fronts recursively:

$$\mathcal{F}_1 = \{\mathbf{x} \in \mathcal{C} : \nexists\, \mathbf{y} \in \mathcal{C},\ \mathbf{y} \prec \mathbf{x}\}, \qquad \mathcal{F}_\ell = \mathrm{ND}\left(\mathcal{C} \setminus \bigcup_{j<\ell} \mathcal{F}_j\right),\ \ell \geq 2, \tag{10}$$

where $\mathbf{x}$ and $\mathbf{y}$ are candidate masks, $\prec$ denotes Pareto dominance, $\mathcal{F}_\ell$ is the ℓ-th front, and $\mathrm{ND}(\cdot)$ returns the non-dominated subset of its input set. Front rank gives a coarse ordering of solution quality: all masks in $\mathcal{F}_1$ are Pareto-competitive within the pool, while later fronts are dominated by earlier ones. When the population budget cuts through a front, NSGA-II sorts candidates in that front by crowding distance in descending order and retains those with larger distances first, with boundary solutions treated as maximally distant. This rule favors solutions located in sparse regions of objective space, preserving coverage of different trade-offs and preventing the population from collapsing to a narrow set of masks that favor only one capability domain.

After search, the final population contains multiple non-dominated masks with different trade-offs, but deployment requires selecting a single pruning decision. SliMOO therefore uses hypervolume

Table 9: Detailed post-pruning results on the Llama and Qwen families. We report benchmark-level scores together with domain averages: lower is better for language modeling perplexity and higher is better elsewhere. **Bold** indicates the best result, and underlines indicate the runner-up.

Model	Sparsity	Method	language modeling				math			code			Knowledge Reasoning							
			Wikitext	PTB	C4	AVG	GSM8K	MATH	AVG	HumanEval	MBPP	AVG	MMLU	ARC-C	ARC-E	SIQA	HellaSwag	PIQA	WinoGrande	AVG
Llama-3.1-70B	0%	N/A	2.63	7.72	6.84	5.73	80.82	24.14	52.48	51.83	64.80	58.32	78.51	60.75	87.12	50.46	66.43	82.86	79.64	72.25
	25% Layer	ShortGPT	9.00	11.47	13.57	11.35	12.74	12.96	12.85	18.29	12.60	15.45	77.79	48.46	76.56	47.95	57.49	76.28	76.95	65.93
		LaCo	8.90	11.60	13.10	11.20	20.77	11.22	16.00	23.78	17.00	20.39	78.21	48.12	76.34	47.59	58.11	76.82	78.37	66.22
		LLM-Pruner	6.12	9.71	10.38	**8.74**	50.64	14.24	32.44	21.95	31.60	26.78	66.72	54.10	82.11	49.23	59.85	80.58	74.90	66.78
		EvoPress	7.16	10.82	11.67	9.88	36.47	12.18	24.33	28.66	41.40	35.03	66.69	50.34	77.40	46.01	57.08	78.78	76.09	64.63
		SliMOO	7.21	10.65	11.81	9.89	53.98	14.82	**34.40**	29.27	49.00	**39.14**	74.12	52.99	79.80	46.98	58.60	78.76	77.03	**66.90**
	33% Sublayer	ShortGPT	38.28	33.54	33.83	35.22	4.25	6.14	5.20	5.49	0.00	2.75	73.00	45.82	72.73	47.49	54.67	73.83	76.09	63.38
		EvoPress	9.31	12.94	13.81	12.02	12.13	8.52	10.33	14.02	16.20	15.11	66.56	44.62	74.54	45.24	53.31	75.46	74.43	62.02
		BlockPruner	7.89	11.66	13.42	10.99	37.38	11.06	24.22	10.98	26.80	18.89	67.77	50.77	79.08	46.52	58.06	78.40	67.09	63.96
		FinerCut	9.33	15.15	14.06	12.85	16.07	10.12	13.10	7.93	7.80	7.87	67.29	50.26	76.26	46.62	58.22	77.69	76.79	64.73
		SliMOO	7.34	11.81	12.19	**10.45**	42.53	12.00	**27.27**	22.56	39.00	**30.78**	71.38	53.41	79.50	46.47	57.77	78.67	76.80	**66.29**
Llama-3.1-405B	0%	N/A	1.36	6.94	5.75	4.68	88.40	32.42	60.41	54.88	71.00	62.94	85.02	63.05	86.78	53.17	70.33	84.00	82.71	75.01
	25% Layer	ShortGPT	5.48	8.85	9.98	8.10	38.67	18.68	28.68	40.85	56.60	48.73	83.81	54.86	81.31	49.07	63.64	80.14	81.22	70.58
		LaCo	5.86	9.27	10.99	8.71	58.07	23.30	40.69	45.12	53.40	49.26	83.66	55.23	82.11	49.52	63.17	80.74	79.72	70.59
		EvoPress	4.22	8.50	9.62	**7.45**	69.45	20.44	44.95	37.80	56.00	46.90	77.63	57.76	82.83	50.15	63.26	81.12	78.77	70.22
		SliMOO	4.22	8.75	9.81	7.59	76.95	22.70	**49.83**	48.78	55.20	**51.99**	81.69	57.00	83.59	49.28	64.12	81.77	79.79	**71.03**
	33% Sublayer	ShortGPT	31.27	16.45	19.50	22.41	3.11	0.00	1.56	0.00	0.00	0.00	22.95	19.97	26.56	34.08	25.88	51.47	50.20	33.02
		EvoPress	5.97	9.81	12.00	9.26	42.61	16.02	29.32	30.49	52.00	41.25	75.82	50.51	79.50	48.26	60.29	79.49	75.85	67.10
		BlockPruner	5.00	8.42	9.18	7.53	66.34	18.64	42.49	28.66	51.60	40.13	78.66	54.95	81.48	54.91	61.65	79.65	73.48	69.25
		FinerCut	6.04	10.72	13.15	9.97	69.45	21.56	45.51	39.02	48.60	43.81	82.54	56.91	82.70	50.77	63.41	80.58	80.98	71.13
		SliMOO	4.09	8.23	9.50	**7.27**	81.50	27.63	**54.57**	56.10	63.80	**59.95**	80.26	60.32	85.02	48.93	65.39	82.70	79.32	**71.71**
Qwen3-32B	0%	N/A	7.19	13.51	11.97	10.89	79.40	33.48	56.44	79.88	64.80	72.34	81.88	58.45	84.13	52.41	63.86	81.28	73.16	70.74
	25% Layer	ShortGPT	12.72	25.37	22.28	20.12	20.70	6.76	13.73	8.54	15.60	12.07	44.22	38.57	70.54	41.20	45.56	72.47	53.99	52.36
		LaCo	10.46	18.76	17.47	15.56	57.62	24.50	41.06	53.05	11.80	32.43	65.18	49.74	79.92	45.80	55.26	74.70	65.35	62.28
		LLM-Pruner	7.68	13.02	13.05	**11.25**	37.38	19.50	28.44	6.10	19.20	12.65	76.64	22.70	25.08	47.29	60.29	49.51	69.69	50.17
		EvoPress	9.70	16.76	15.89	14.12	73.69	22.30	48.00	39.02	54.80	46.91	69.08	52.73	79.71	46.11	53.85	75.46	65.11	63.15
		SliMOO	9.62	16.62	15.62	13.95	75.89	24.40	**50.15**	57.32	56.80	**57.06**	68.21	53.58	79.29	46.57	53.71	75.35	66.30	**63.29**
	33% Sublayer	ShortGPT	13.51	26.10	24.28	21.30	45.79	16.72	31.26	42.68	27.80	35.24	44.88	42.41	70.50	42.22	45.66	69.70	56.91	53.18
		EvoPress	9.81	18.62	16.88	**15.10**	56.10	15.06	35.58	18.90	37.60	28.25	57.79	42.32	71.84	41.97	50.79	73.12	58.64	56.64
		BlockPruner	12.32	23.17	19.81	18.43	44.43	14.34	29.39	19.51	30.20	24.86	50.56	43.09	73.78	42.78	48.63	72.85	60.85	56.08
		FinerCut	19.49	43.06	26.76	29.77	1.30	1.04	1.17	3.05	4.00	3.53	46.02	33.87	57.41	42.53	46.99	71.16	55.64	50.52
		SliMOO	10.75	20.75	18.00	16.50	58.30	20.64	**39.47**	46.34	46.00	**46.17**	65.90	47.27	76.77	44.42	51.19	75.14	62.67	**60.48**
Qwen3-235B-A22B	0%	N/A	5.63	11.58	10.39	9.20	84.99	42.48	63.74	79.27	79.60	79.44	86.35	61.86	85.27	54.96	67.88	81.99	78.93	73.89
	25% Layer	ShortGPT	6.07	12.16	11.83	10.02	84.23	35.82	60.03	31.71	74.40	53.06	79.33	56.91	83.04	50.15	62.78	78.94	73.80	69.28
		LaCo	6.22	13.37	11.60	10.40	81.80	36.56	59.18	31.10	71.80	51.45	79.69	57.68	83.21	50.87	63.40	79.22	74.90	69.85
		EvoPress	5.72	11.44	10.94	**9.37**	82.64	38.26	60.45	34.76	74.80	54.78	82.38	58.28	82.53	52.81	64.93	80.69	75.77	71.06
		SliMOO	5.72	11.44	11.06	9.41	84.38	40.00	**62.19**	35.98	76.80	**56.39**	85.53	62.97	85.44	52.87	65.12	80.41	76.80	**72.73**
	33% Sublayer	ShortGPT	6.18	12.35	12.23	10.25	80.82	36.30	58.56	35.98	71.80	53.89	77.47	53.67	81.10	49.54	61.74	80.09	73.88	68.21
		EvoPress	5.93	11.67	11.02	**9.54**	72.86	25.18	49.02	26.22	52.20	39.21	75.44	54.44	80.93	51.79	59.92	79.33	72.61	67.78
		BlockPruner	7.37	12.95	12.67	11.00	60.05	25.10	42.58	37.80	51.20	44.50	73.46	53.67	81.10	50.05	59.63	78.56	71.35	66.83
		FinerCut	6.75	14.98	12.40	11.38	65.35	26.38	45.87	31.71	43.40	37.56	74.06	54.27	79.00	48.87	60.47	78.78	71.43	66.70
		SliMOO	5.94	11.81	11.81	9.85	80.67	38.70	**59.69**	39.63	74.40	**57.02**	82.57	59.64	83.84	50.26	63.08	80.03	74.35	**70.54**

contribution to choose the final mask. Let $\mathcal{P}$ denote the final Pareto population and set the reference point by $r_t = 1.1 \max_{\mathbf{x}\in\mathcal{A}} f_t(\mathbf{x})$, where $\mathcal{A}$ is the highest-fidelity evaluated archive. For minimization objectives, the hypervolume is the dominated volume:

$$\mathrm{HV}(\mathcal{P}; \mathbf{r}) = \lambda\left(\bigcup_{\mathbf{x}\in\mathcal{P}} [\mathbf{f}(\mathbf{x}), \mathbf{r}]\right), \tag{11}$$

where $\lambda(\cdot)$ denotes volume and $[\mathbf{f}(\mathbf{x}), \mathbf{r}]$ is the axis-aligned box from candidate $\mathbf{x}$ to the reference point. The marginal contribution of a candidate $\mathbf{x} \in \mathcal{P}$ is:

$$\Delta_{\mathrm{HV}}(\mathbf{x}; \mathcal{P}) = \mathrm{HV}(\mathcal{P}; \mathbf{r}) - \mathrm{HV}(\mathcal{P} \setminus \{\mathbf{x}\}; \mathbf{r}). \tag{12}$$

A large contribution indicates that the candidate expands the covered Pareto region rather than merely duplicating nearby trade-offs. Thus, hypervolume provides a principled tie-breaker for selecting a compact final mask from a diverse Pareto population. Overall, the derivation shows how SliMOO connects three levels of reasoning: TLIP provides a first-order task-layer prior, NSGA-II converts task-wise losses into diversity-preserving Pareto selection, and hypervolume identifies a final solution with strong global contribution to the trade-off surface.

Multi-stage evaluation for efficient search. To avoid evaluating every candidate with the full calibration budget, we use a successive schedule inside the objective evaluation step of Algorithm 1. Candidate pools are filtered by NSGA-II with token budgets $2{,}048 \rightarrow 4{,}096 \rightarrow 16{,}384$ and survivor counts $16 \rightarrow 8 \rightarrow 4$; the same schedule is used for initialization. This accelerates search by discarding weak masks at low fidelity while leaving candidate generation and Pareto selection unchanged.

A.2 Detailed Post-pruning Performance

Table 9 expands the domain averages in the main text into a benchmark-level view, making it easier to distinguish broad capability preservation from gains concentrated on a single metric. Across all models, SliMOO is consistently strongest on the math and code averages while remaining competitive on language modeling and knowledge reasoning. This pattern is similarly clear on Qwen family models, where the benchmark-level results show gains across multiple math and code benchmarks rather than a single outlier score.

Consistent with this observation, on Qwen3-32B at 25% sparsity, relative to EvoPress, SliMOO improves GSM8K/MATH from $73.69\%/22.30\%$ to $75.89\%/24.40\%$ and HumanEval/MBPP from $39.02\%/54.80\%$ to $57.32\%/56.80\%$, while slightly improving the language modeling average from 14.12 to 13.95. The same trend is even stronger on Qwen3-235B-A22B at 33% sparsity, where SliMOO improves GSM8K/MATH from $72.86\%/25.18\%$ to $80.67\%/38.70\%$ and HumanEval/MBPP from $26.22\%/52.20\%$ to $39.63\%/74.40\%$, while also raising the knowledge reasoning average from 67.78% to 70.54%; the Llama experiments show the same direction with gains across all domains. Together, these results support the paper's central claim that task-aware, Pareto-guided pruning preserves broad capability more reliably than a single global pruning order.

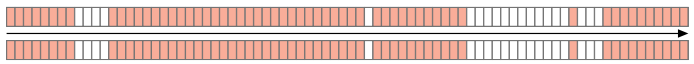
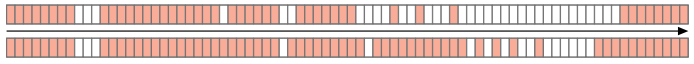
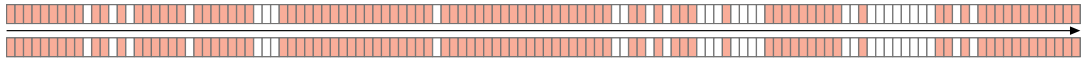
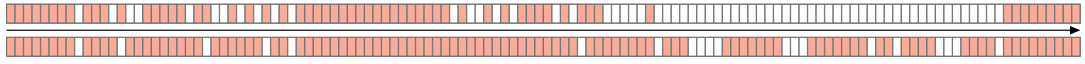
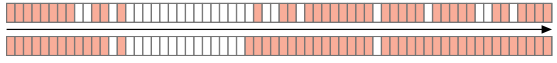
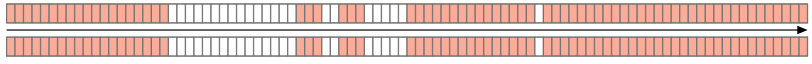
A.3 Pruning and Tuning

Post-pruning fine-tuning is commonly used to recover compression-induced capability loss with controllable additional cost. Table 11 evaluates this setting with lightweight LoRA tuning [55] after depth pruning. Starting from the pruned Llama-3.1-70B checkpoints, we tune all linear layers with LoRA rank 16, $\alpha = 32$, dropout 0.05, learning rate 1.0×10^{-4}, cosine scheduling, and warmup ratio 0.03. Training uses roughly 25% of `tulu-3-sft-mixture`, a broad supervised instruction-tuning mixture containing about 939K examples from diverse sources, including general instruction following, math, code, multilingual, and domain-specific data [56]. Finetuning improves all pruned models, but SliMOO remains the strongest: SliMOO*

Table 11: Post-pruning LoRA tuning results on Llama-3.1-70B at 25% layer sparsity. Starred methods are tuned after pruning with LoRA.

Method	Lang. ↓	Math ↑	Code ↑	Know. ↑
ShortGPT	11.35	12.85	15.45	65.93
ShortGPT*	9.13	37.35	49.18	68.31
EvoPress	9.88	24.33	35.03	64.63
EvoPress*	**8.48**	42.22	57.23	68.32
SliMOO	9.89	34.40	39.14	66.90
SliMOO*	8.56	**48.04**	**61.40**	**70.23**

Table 10: Layer-wise pruning configurations selected by SliMOO. Each column denotes one transformer layer, with attention on the top strip and MLP on the bottom strip. Salmon blocks are retained and white blocks are pruned; arrows point from shallow to deep layers.

Model	Sparsity	Mask configuration
Llama-3.1-70B	25% Layer	
	33% Sublayer	
Llama-3.1-405B	25% Layer	
	33% Sublayer	
Qwen3-32B	25% Layer	
	33% Sublayer	
Qwen3-235B-A22B	25% Layer	
	33% Sublayer	



achieves the best math, code, and knowledge averages while matching the best language-modeling recovery closely. This suggests that SliMOO's pruning mask preserves a more tunable residual model than masks selected by single-score or scalarized search baselines.

A.4 Small Model Experiments

As model scale increases, depth pruning can exploit parameter redundancy that is less pronounced in smaller models. We therefore additionally evaluate Llama-3.1-8B as a small-model setting, where pruning is expected to be more fragile and use milder budgets: 12.5% whole-layer pruning and 25% attention/MLP sublayer pruning. Table 12 shows that SliMOO remains effective in this more sensitive regime. Under 12.5% layer sparsity, SliMOO achieves the best pruned result on all four domain averages, indicating that task-aware search can identify removable layers even when redundancy is limited. Under 25% sublayer pruning, SliMOO obtains the best math, code, and knowledge averages and remains competitive on language modeling, while several baselines suffer severe capability collapse. These results suggest that SliMOO's multi-objective formulation is useful not only for highly redundant large models but also for smaller models where pruning decisions must be more conservative.

Table 12: Small-model pruning results on Llama-3.1-8B. We report domain averages: lower is better for language modeling and higher is better elsewhere. **Bold** marks the best pruned model within each sparsity setting.

Sparsity	Method	Lang. ↓	Math ↑	Code ↑	Know. ↑
0%	N/A	8.47	32.10	42.20	66.04
12.5% Layer	ShortGPT	17.60	19.17	8.37	61.28
	LaCo	12.73	21.02	17.93	60.13
	EvoPress	19.54	1.27	3.02	48.40
	SliMOO	**11.65**	**23.30**	**23.26**	**61.64**
25% Sublayer	ShortGPT	280.00	0.02	0.00	46.89
	EvoPress	24.48	1.68	2.17	47.32
	BlockPruner	15.94	2.80	3.65	54.63
	FinerCut	**15.12**	5.30	5.74	53.73
	SliMOO	15.58	**8.46**	**9.47**	**55.04**

A.5 Pruning Position and Interpretability

Table 10 visualizes the final pruning decisions selected by SliMOO across model families and pruning granularities. TLIP provides a task-aware prior that biases search away from task-critical units, while Pareto evaluation determines which combinations are actually retained. This makes the final masks interpretable as search-refined versions of the task-layer importance patterns in Figure 2(a). For example, under 33% sublayer pruning on Llama-3.1-70B, SliMOO keeps most early and final sublayers, but prunes a concentrated mid-to-late region, especially attention sublayers from roughly layers 41–71, with selected MLP removals interleaved around layers 42 and 54–68. This pattern is consistent with TLIP: broadly important units are protected, while lower-sensitivity or task-redundant regions provide pruning capacity. At the same time, the mask is not purely local or monotonic, reflecting the role of Pareto search in accounting for cross-task interactions among removed sublayers.

A.6 Robustness to Search Random Seeds

We assess the sensitivity of SliMOO to search randomness on Llama-3.1-70B. At 25% layer sparsity, all three runs converge to the same pruning mask and yield identical domain averages: 9.89 perplexity for language modeling and 34.40%, 39.14%, and 66.90% for math, code, and knowledge reasoning, respectively. At 33% sublayer sparsity, we evaluate five search seeds, including the reported run. As shown in Table 13, the domain averages exhibit limited variation across runs. These results support the robustness of SliMOO to search randomness at both pruning granularities in the evaluated setting.

Table 13: Search-seed stability on Llama-3.1-70B at 33% sublayer sparsity. The last column reports mean $\pm$ SD across five runs. Language reports perplexity; other scores are percentages.

Domain	EvoPress	SliMOO reported	SliMOO 5 runs
Language ↓	12.02	10.45	10.34 ± 0.14
Math ↑	10.33	27.27	27.31 ± 0.17
Code ↑	15.11	30.78	30.56 ± 0.45
Knowledge ↑	62.02	66.29	66.19 ± 0.07

Table 14: Case study on Llama-3.1-70B before and after dropping 25% of layers with SliMOO. We use greedy decoding for generation. For math, code, and reasoning, decoding is truncated after the final answer.

Domain	Prompt (truncated)	Dense output	Pruned output
Language modeling	*"The Great Wall of China is a series of fortifications that were built across the historical northern borders of ancient Chinese states and Imperial China as protection against various nomadic groups. Construction of the walls began as early as the 7th century BC, and"*	they have a total length of 13,000 miles. Several sections of the wall are now in ruins or have disappeared. However, the wall is still one of the most attractive places in the world and was selected as one of the New Seven Wonders of the World in 2007.	the most recent construction was completed in the 18th century. The walls were constructed of stone, compacted earth, and compacted sand, depending on the area. The walls were constructed on high ground, and they overlook nearby plains. The walls were constructed with the idea of protecting the states from raiders, and they were constructed with the assumption that they could be defended by a minimal force.
Math	*"Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for $2 per fresh duck egg. How much in dollars does she make every day at the farmers' market?"*	Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for $2 per fresh duck egg. $16 - 3 - 4 = 9$. $9 \times 2 = 18$. **The answer is $18.** ✓	Janet's ducks lay $16 - 3 - 4 = 9$ eggs per day. She sells 9 eggs per day. She makes $9 \times 2 = \$18$ per day. **The answer is $18.** ✓
Code	`def is_prime(n: int) -> bool: """Return True if n is prime."""`	`if n < 2: return False for i in range(2,\ int(n ** 0.5) + 1): if n % i == 0: return False return True` **Optimal $O(\sqrt{n})$.** ✓	`if n < 2: return False for i in range(2, n): if n % i == 0: return False return True` Functionally correct but **naive** $O(n)$; the $\sqrt{n}$ optimization is lost. ✓
Knowledge reasoning	*"In which year did World War II end? A. 1943 B. 1944 C. 1945 D. 1946 Answer:"*	**C** ✓	**C** ✓

A.7 Case Study

Table 14 provides a qualitative comparison between the dense and SliMOO-pruned Llama-3.1-70B on representative prompts from the four capability domains. Overall, both models produce strong outputs, indicating that the pruned model retains the core capabilities of the dense model after removing 25% of the layers. The differences are mostly subtle: in language modeling, the pruned continuation remains fluent and coherent; in math and knowledge reasoning, it preserves the correct final answer; and in code, it remains functionally correct although less algorithmically refined than the dense model. These examples qualitatively reinforce the main finding of the paper that SliMOO preserves broad capability across diverse domains after depth pruning.

A.8 Existing Assets and Licenses

Table 15 lists the main existing assets used in our experiments and the licenses reported by their source pages.

A.9 Broader Impacts

SliMOO has primarily positive broader impacts by making large language models more efficient to deploy. By reducing model depth while preserving broad capabilities, it can lower inference

Table 15: Existing models, datasets, and benchmarks used in the paper.

Asset	License	Source
Llama-3.1 [2]	Llama 3.1 Community	https://llama.com/llama3_1/license/
Qwen3 [3]	Apache-2.0	https://huggingface.co/Qwen/Qwen3-32B/blob/main/LICENSE; https://huggingface.co/Qwen/Qwen3-235B-A22B/blob/main/LICENSE
FineWeb-Edu [57]	ODC-BY-1.0	https://huggingface.co/datasets/HuggingFaceFW/fineweb-edu
FineWeb-2 [54]	ODC-BY-1.0	https://huggingface.co/datasets/HuggingFaceFW/fineweb-2
GSM8K [30]	MIT	https://huggingface.co/datasets/openai/gsm8k
DeepScaleR [31]	MIT	https://huggingface.co/datasets/agentica-org/DeepScaleR-Preview-Dataset
python-codes-25k [32]	MIT	https://huggingface.co/datasets/flytech/python-codes-25k
commonsense_170k [29]	Apache-2.0	https://huggingface.co/datasets/zwhe99/commonsense_170k; https://github.com/AGI-Edgerunners/LLM-Adapters
Tulu-3 SFT mixture [56]	ODC-BY-1.0	https://huggingface.co/datasets/allenai/tulu-3-sft-mixture
WikiText-2 [34]	CC-BY-SA-3.0 / GFDL	https://huggingface.co/datasets/Salesforce/wikitext
PTB [35]	LDC User Agreement	https://catalog.ldc.upenn.edu/LDC99T42
C4 [36]	ODC-BY-1.0	https://huggingface.co/datasets/allenai/c4
MATH [37]	MIT	https://github.com/hendrycks/math
HumanEval [38]	MIT	https://github.com/openai/human-eval
MBPP [39]	CC-BY-4.0	https://huggingface.co/datasets/google-research-datasets/mbpp
MMLU [40]	MIT	https://huggingface.co/datasets/cais/mmlu
ARC-C/E [41]	CC-BY-SA-4.0	https://huggingface.co/datasets/allenai/ai2_arc
Social-IQA [42]	CC-BY-4.0	https://huggingface.co/datasets/allenai/social_i_qa
HellaSwag [43]	MIT	https://github.com/rowanz/hellaswag
PIQA [44]	CC-BY-SA-4.0	https://huggingface.co/datasets/ybisk/piqa
WinoGrande [45]	Apache-2.0	https://github.com/allenai/winogrande
RULER (NIAH-8K) [46]	Apache-2.0	https://github.com/NVIDIA/RULER
TruthfulQA [47]	Apache-2.0	https://huggingface.co/datasets/truthfulqa/truthful_qa
CrowS-Pairs [48]	CC-BY-SA-4.0	https://github.com/nyu-mll/crows-pairs
ToxiGen [49]	CDLA-Permissive-2.0	https://huggingface.co/datasets/toxigen/toxigen-data; https://github.com/microsoft/TOXIGEN/blob/main/LICENSE.txt
MedMCQA [50]	Apache-2.0	https://huggingface.co/datasets/openlifescienceai/medmcqa

latency, memory footprint, and serving cost, making capable LLMs more accessible under practical resource constraints. These efficiency gains may also reduce the energy and hardware requirements of deployment. SliMOO does not introduce new data, new model capabilities, or new high-risk modalities beyond the scope of the pre-trained models it compresses. However, pruning may affect safety guardrails or instruction alignment unevenly, so deployment should evaluate relevant safety and alignment benchmarks when these properties are required.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the main contributions of SliMOO, including TLIP and Pareto-aware evolutionary search, and the experimental claims are supported by results in Section 4 and Appendix A.2.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations are discussed in Section 5, including dependence on predefined calibration domains, open-ended deployment settings, and the offline cost of TLIP profiling and search.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper does not claim formal theoretical theorems. It provides algorithmic definitions and a detailed derivation of SliMOO in Appendix A.1.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Section 4.1 reports the models, pruning granularities, calibration data, evaluation tasks, implementation details, and search hyperparameters. Additional benchmark-level results and stability information are provided in the appendix.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The code has not yet been publicly released and will be made available at `https://github.com/AMD-AGI/SliMOO`. The paper describes the method and experimental settings and provides the sources and licenses of the existing models, datasets, and software used in our experiments.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The experimental setup specifies calibration sources and token budgets, response-only masking for math and code, evaluation benchmarks and shot settings, baselines, software stack, precision, and search hyperparameters.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Appendix A.6 reports a run-to-run stability study on Llama-3.1-70B at 25% layer sparsity, where three repeated runs converge to the same pruning mask and identical domain averages. We use this stability check rather than standard error bars because the output is a discrete pruning mask.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The Efficiency Analysis reports the AMD Instinct MI300X GPU configuration, memory budgets, profiling time, and per-generation search time for 70B and 405B experiments, as well as post-pruning inference efficiency.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work studies compression of existing LLMs using publicly available datasets and benchmarks, and we are not aware of any deviation from the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The broader impacts are discussed in Appendix A.9. The work has positive impact by improving the efficiency of LLM deployment and reducing serving cost, while potential misuse risks inherit those of the underlying models rather than being unique to SliMOO.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release a new pretrained model, dataset, or high-risk generative asset. It proposes a pruning framework evaluated on existing LLMs and benchmarks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The paper cites the pretrained models, datasets, and evaluation benchmarks used in the experiments; Appendix A.8 lists their licenses and source links. We use these assets according to their intended research usage and applicable terms.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification: The paper does not introduce or release a new dataset, benchmark, or pretrained model asset. It introduces an algorithmic pruning framework and reports experimental results.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The work does not involve crowdsourcing experiments or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The work does not involve human subjects and therefore does not require IRB approval.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.